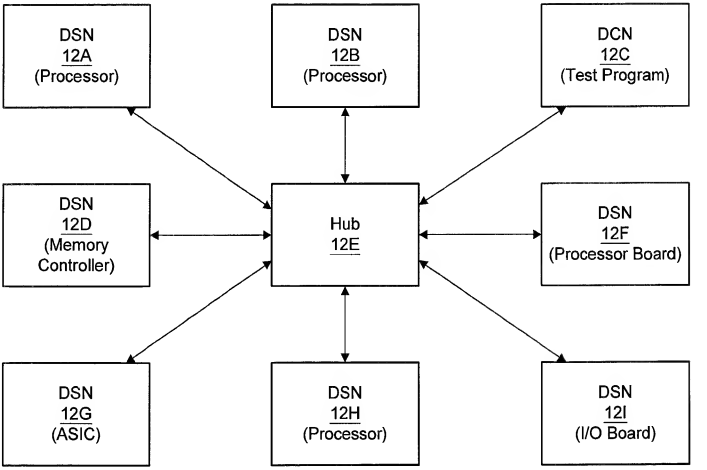
DSN
12A
(Processor)
DSN
12B
(Processor)
DCN
12C
(Test Program)
DSN
12D
(Memory
Controller)
Hub
12E
DSN
12F
(Processor Board)
DSN
12G
(ASIC)
DSN
12H
(Processor)
DSN
12I
(I/O Board)
10

Fig. 1

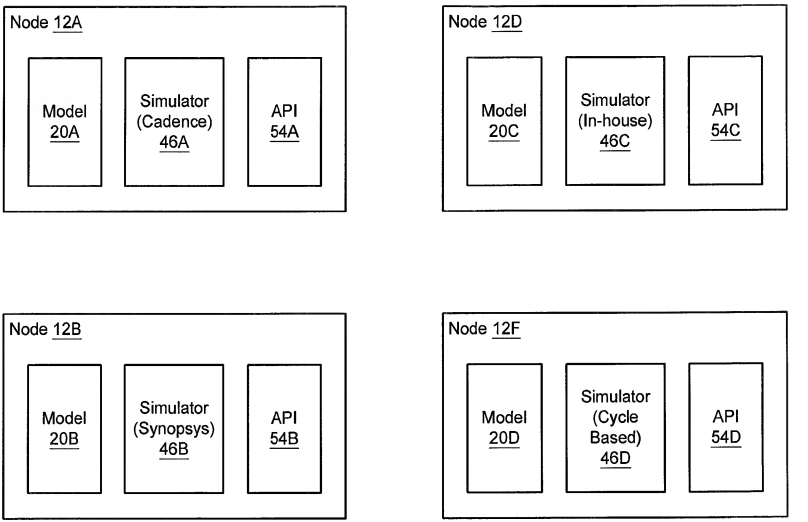

Fig. 2

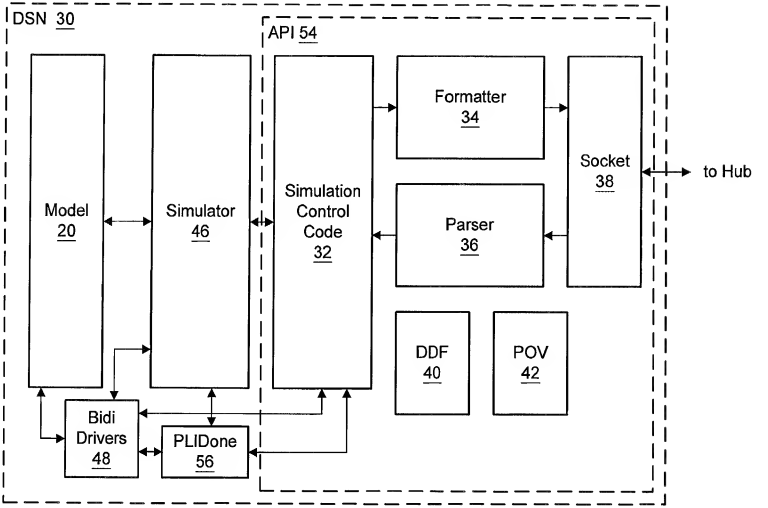
DSN 30
API 54
Model 20
Simulator 46
Simulation Control Code 32
Formatter 34
Parser 36
Socket 38
to Hub
DDF 40
POV 42
Bidi Drivers 48
PLIDone 56

Fig. 3

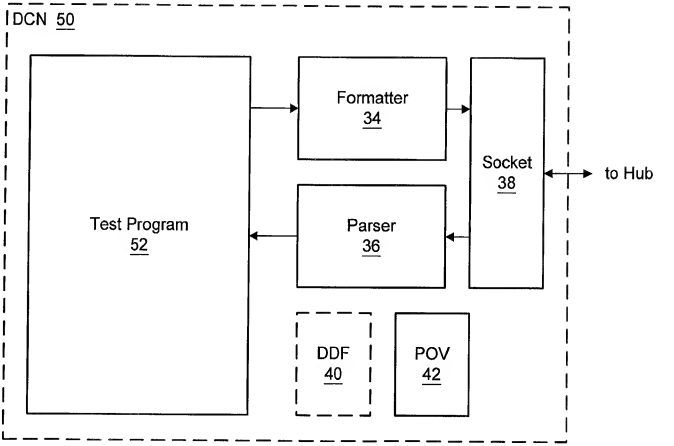

Fig. 4

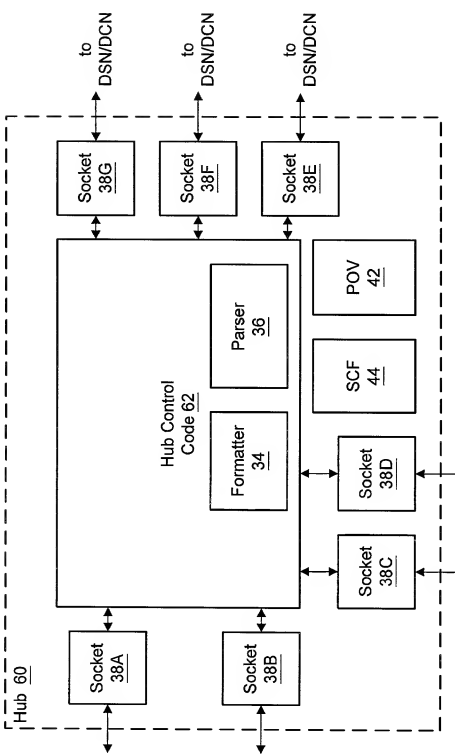
Hub 60
Socket 38A
Socket 38B
Socket 38C
Socket 38D
Socket 38E
Socket 38F
Socket 38G
Hub Control Code 62
Formatter 34
Parser 36
SCF 44
POV 42
to DSN/DCN
to DSN/DCN
to DSN/DCN
to DSN/DCN
to DSN/DCN
to DSN/DCN
to DSN/DCN

Fig. 5

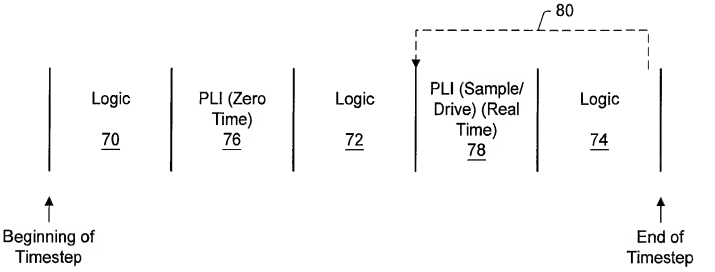
80
Logic
70
PLI (Zero Time)
76
Logic
72
PLI (Sample/ Drive) (Real Time)
78
Logic
74
Beginning of Timestep
End of Timestep

Fig. 6

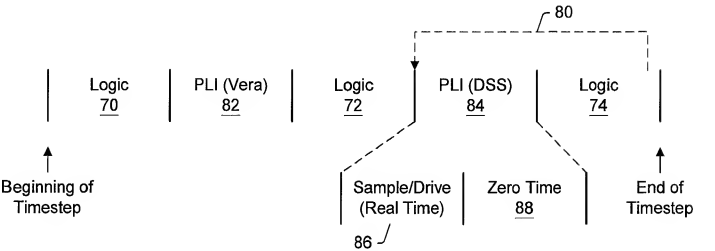
80
Logic
70
PLI (Vera)
82
Logic
72
PLI (DSS)
84
Logic
74
Beginning of Timestep
Sample/Drive (Real Time)
86
Zero Time
88
End of Timestep

Fig. 7

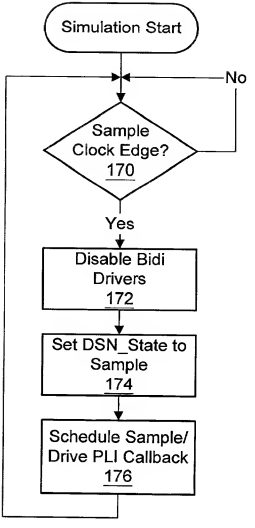
Simulation Start
No
Sample Clock Edge? 170
Yes
Disable Bidi Drivers 172
Set DSN_State to Sample 174
Schedule Sample/ Drive PLI Callback 176

Fig. 8

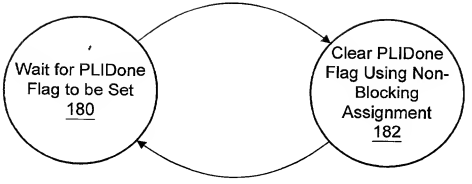
Wait for PLIDone Flag to be Set 180
Clear PLIDone Flag Using Non-Blocking Assignment 182

Fig. 9

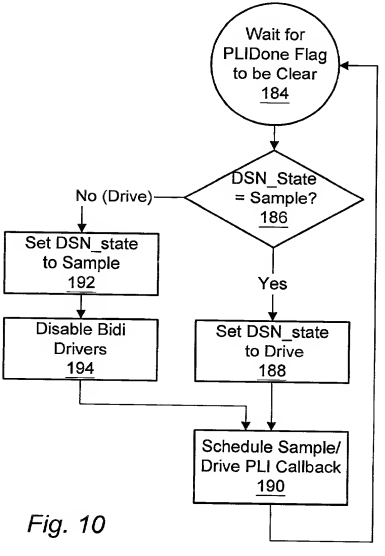
Wait for PLIDone Flag to be Clear 184
DSN_State = Sample? 186
No (Drive)
Yes
Set DSN_state to Sample 192
Disable Bidi Drivers 194
Set DSN_state to Drive 188
Schedule Sample/ Drive PLI Callback 190

Fig. 10

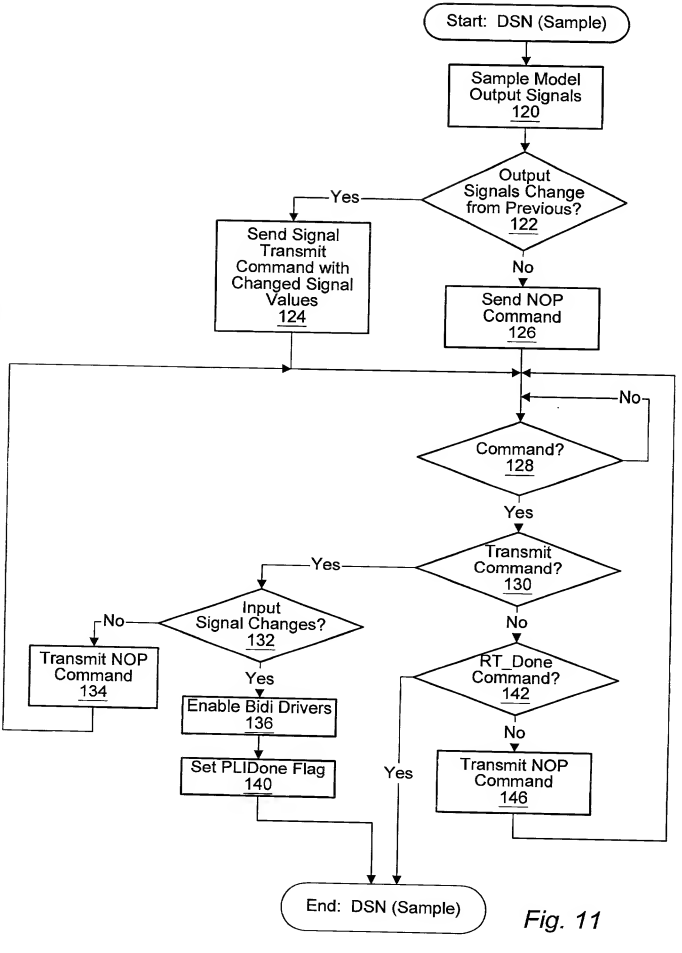
Start: DSN (Sample)
Sample Model Output Signals 120
Output Signals Change from Previous? 122
Yes
Send Signal Transmit Command with Changed Signal Values 124
No
Send NOP Command 126
No
Command? 128
Yes
Transmit Command? 130
Yes
Input Signal Changes? 132
No
Transmit NOP Command 134
Yes
Enable Bidi Drivers 136
Set PLIDone Flag 140
No
RT_Done Command? 142
Yes
No
Transmit NOP Command 146
End: DSN (Sample)

Fig. 11

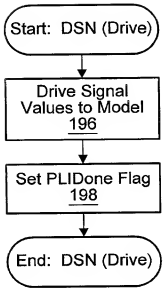

Fig. 12

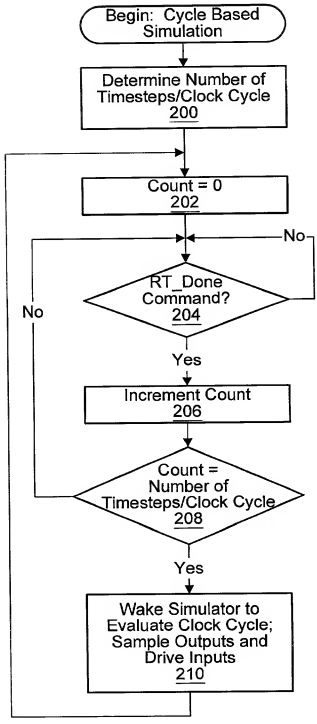

Fig. 13

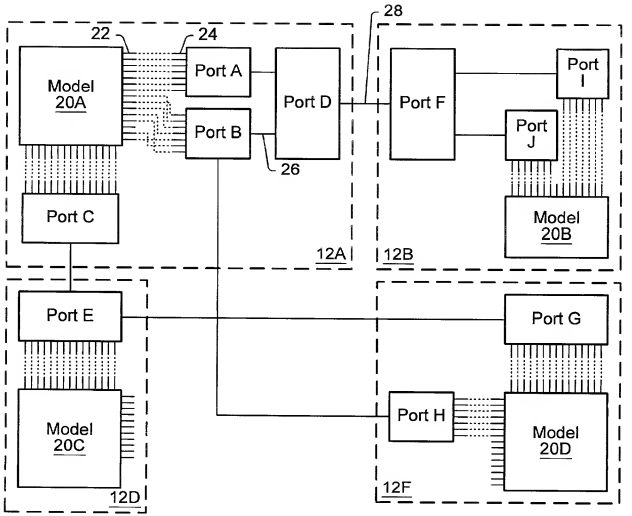
22
24
28
Model
20A
Port A
Port B
Port D
26
Port F
Port
I
Port
J
Model
20B
Port C
12A
12B
Port E
Port G
Model
20C
Port H
Model
20D
12D
12F

Fig. 14

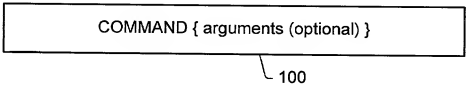
COMMAND { arguments (optional) }
100

Fig. 15

Command	Arguments
POV	port definitions
SCF	model instances and routing
DDF	logical signal to physical signal mapping
TRANSMIT	source model instance, chip names, port names, signalnames, values, and strengths
NOP	source model instance
RT_DONE	source model instance
ZT_DONE	source model instance
ZT_FINISH	source model instance
FINISH	source model instance
USER	source model instance, user message
ERROR	source model instance, error message
HOTPLUG	source model instance, destination model instance
HOTPULL	source model instance, destination model instance
STOP	source model instance

Fig. 16

```
pov : POV '{' portdefs '}'
    ;
portdefs : | pordefs porttype '{' port_member more_members '}' ';'
         ;
porttype : PORTWORD BASENAME
         ;
port_member : SIGNALWORD NAME ';' | BASENAME NAME ';'
            ;
more_members : | more_members port_member
             ;
```

Fig. 17

```
scf : SCF '{' scf_description '}'
    ;
scf_description : | model_instances routing
                ;
model_instances : one_instance more_instances
                ;
one_instance : BASENAME NAME ';'
             ;
more_instances : | more_instances one_instance
               ;
routing : | routing routing_exp
        ;
routing_exp : SCOPENAME1 '->' SCOPENAME1 ';'
            ;
```

Fig. 18

```
ddf : DDF '{' models '}'
    ;
models : | one_model more_models ';'
       ;
one_model : BASENAME '{' log_phys '}' ';'
          ;
more models : | one model more models
            ;
log_phys : | logical physical
         ;
logical : LOGICAL '{' log_defs '}' ';'
        ;
log_defs : | BASENAME NAME ';'
         ;
physical : PHYSICAL '{' phys_defs '}' ';'
         ;
phys_defs : | one_to_one | many_to_one | one_to_many
          ;
one_to_one : SIGNALWORD signalname '{' signalparts '}' ';'
           | one_to_one SIGNALWORD signalname '{' signalparts '}' ';'
           ;
signalname : BITWIDTH BASENAME | BASENAME
           ;
signalparts : one_signalpart more_signalparts
            ;
one_signalpart : NAME '=' SCOPENAME2 ';'
               ;
more_signalparts : | more_signalparts one_signalpart
                 ;
many_to_one : FORALL SIGNALWORD '(' sign_names ')' '{' logicalname '}' ';'
            ;
sig_names : signalname | sig_names ',' signalname
          ;
logicalname : SCOPENAME2 ';'
            ;
one_to_many : FOR SIGNALWORD '(' signalname ')' '{' signalparts '}' ';'
            ;
```

Fig. 19

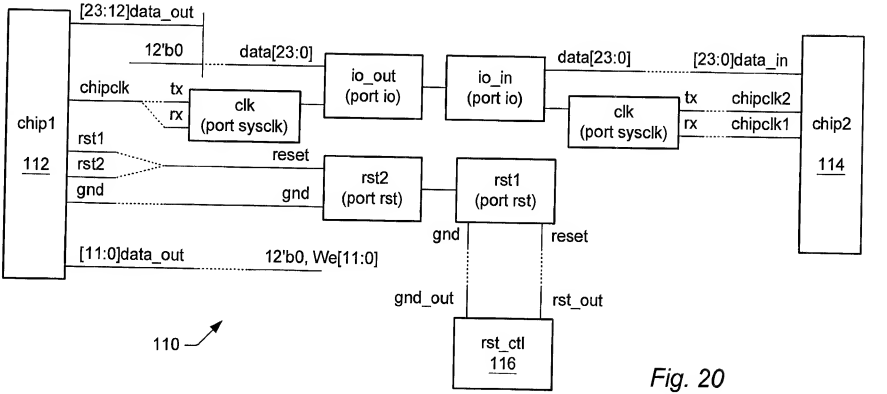
[23:12]data_out
12'b0
data[23:0]
io_out
(port io)
io_in
(port io)
data[23:0]
[23:0]data_in
chipclk
tx
rx
clk
(port sysclk)
clk
(port sysclk)
tx
rx
chipclk2
chipclk1
chip1
112
chip2
114
rst1
rst2
gnd
reset
gnd
rst2
(port rst)
rst1
(port rst)
gnd
reset
[11:0]data_out
12'b0, We[11:0]
gnd_out
rst_out
rst_ctl
116
110

Fig. 20

```
POV {
        port io {
                signal data[23:0];
                sysclk clk;
                };
        port sysclk {
                signal tx;
                signal rx;
                };
        port  rst {
                signal reset;
                signal gnd;
                };
        }
```

Fig. 21

```
SCF {
        chip1 dsn1;
        chip2 dsn2;
        rst_ctl dsn3;

        dsn1.io_out -> dsn2.io_in;
        dsn3.rst1 -> dsn1.rst2;
        }
```

Fig. 22

```
DDF {
  chip1 {
    logical {
      io io_out;
      rst rst2;
    };
    physical {
      signal [23:0] data_out {
            data_out[23:12] = io_out.data[23:12];
            data_out[11:0] ={value: 12'b0, strength: We[11:0] };
       };
       signal gnd { gnd = rst2.gnd };
       for signal (chipclk) {
             chipclk = io_out.clk.tx;
             chipclk = io_out.clk.rx;
       };
       forall signals (rst1, rst2) { rst2.reset };
       signal default_logical { value: 12b'0 } = io_out.data[11:0]; };
    };
  };
}
```

Fig. 23

```
DDF {
  chip2 {
    logical {
      io io_in;
    };
    physical {
      signal [23:0] data_in {
            data_in[23:0] = io_in.data[23:0];
      };
      signal (chipclk1) {
            chipclk1 = io_in.clk.rx;
      };
      signal (chipclk2) {
            chipclk2 = io_in.clk.tx;
      };
    };
  };
}
```

Fig. 24

```
DDF {
  rst_ctl {
    logical {
      rst rst1;
    };
    physical {
      signal rst_out { rst_out = rst1.reset; };
      signal gnd_out { gnd_out = rst1.gnd; };
    };
  };
}
```

Fig. 25

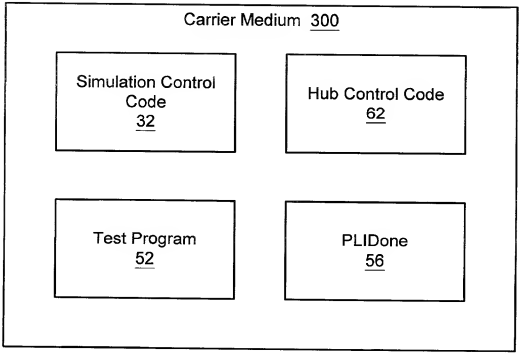

Fig. 26